UTEC
Ciencias de la Computación
CS2702 – Base de Datos 2
Laboratorio 3.2

Fecha: 12/04/2019

Extendible Hash Index & B+ Tree

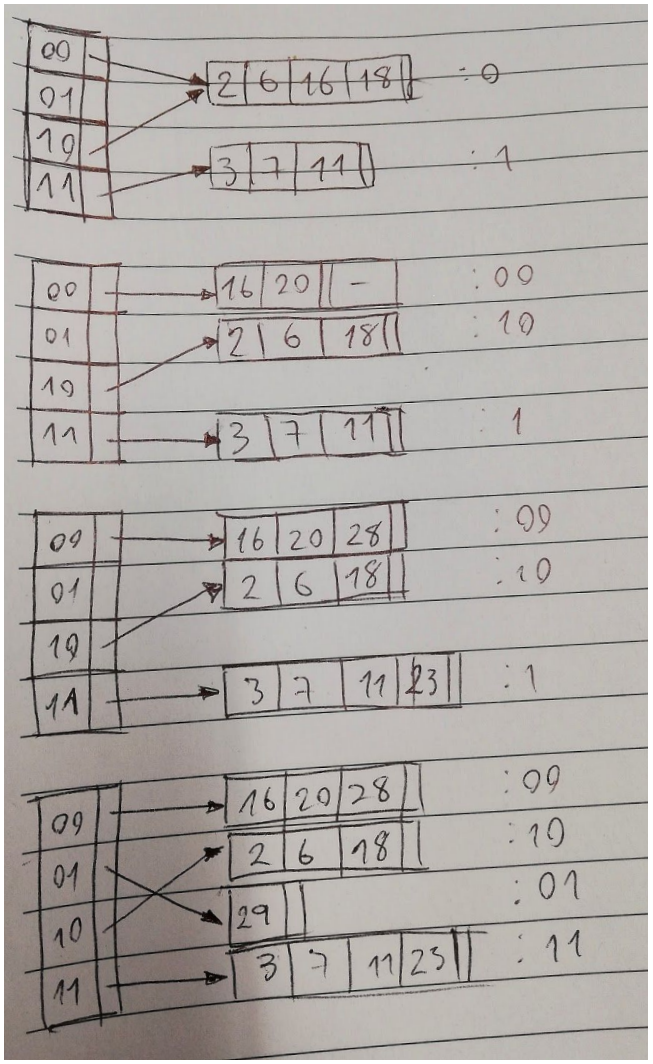
Profesor Heider Sanchez

P1. Extendible Hash:

Dado el siguiente conjunto de claves de registros a insertar en un archivo gestionado por un hash extensible, ¿Cómo quedaría organizado los datos al final de todas las inserciones?

Keys
2,3,6
7,11,16
18,20,23
28,29,30

Considere como profundidad global de dos bits $(d = 2)$ y un factor de bloque de 4 registros $(fb = 4)$. Ilustre paso a paso el proceso de spliting. Sea ordenado y claro en su solución.



Paso 1:

00	**16 20 28**
01	**29**
10	**2 6 18 30**
11	**3 7 11 23**

Luego diseñe el algoritmo de inserción en memoria secundaria.

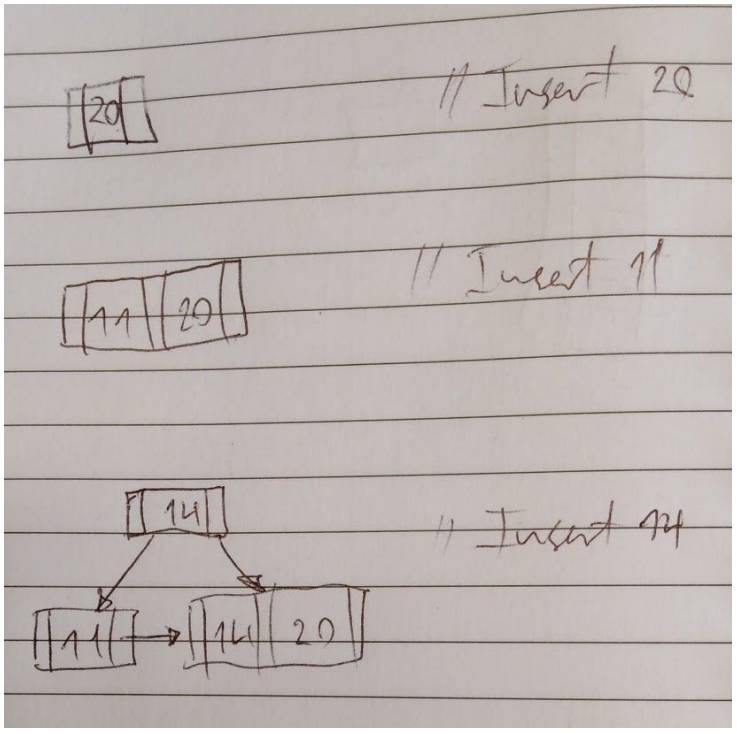
```
insertRegister(register) {
        bucket = getBucket(HashMap[register.key]);
        while (bucket.isFull()) {
                if (bucket.d < D) {    // D es la profundidad global
                        bucket.d++;
                        auxiliarBucket = newBucket();
                        bucket.redistributeRegistersByNewDepth(auxiliarBucket);
                        for b in [bucket, auxiliarBucket] {
                                for mapedIndex in b.mapexIndexes {
                                        HashMap[mapedIndex] = bucket;
                                }
                        }
                }
                else {
                        bucket.next = newBucket();
                        bucket = bucket.next;
                }
        }
        bucket.insert(register);
}
```

P2. B+ Tree: Inserción

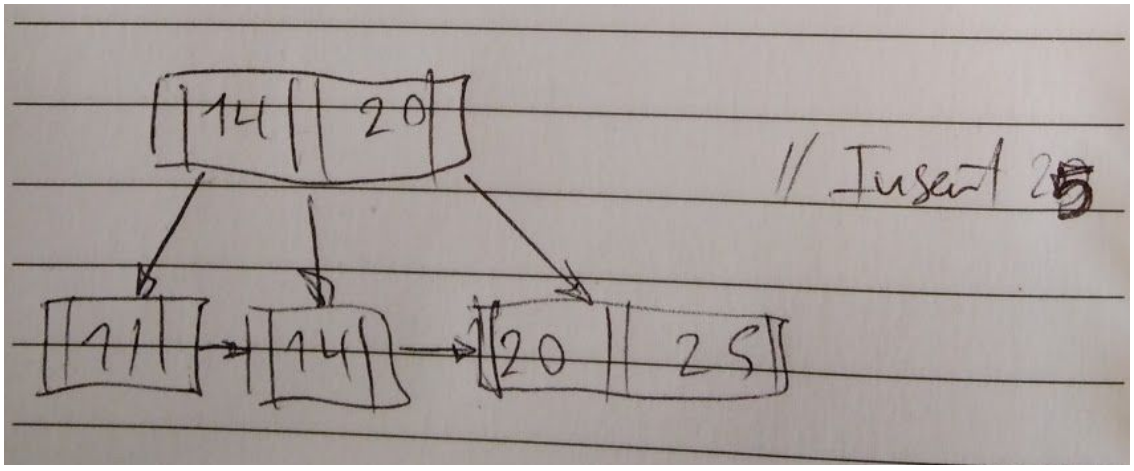
Las inserciones en el árbol B+ deben garantizar la propiedad de balanceo, por lo tanto, los nodos deben tener al menos $fb/2$ entradas (fb es el máximo número de entradas en un nodo). En el

ejercicio siguiente realice las operaciones de inserción aplicando división cuando un nodo esta lleno. Considerar $fb = 3$, $fb_{leaf} = 2$.

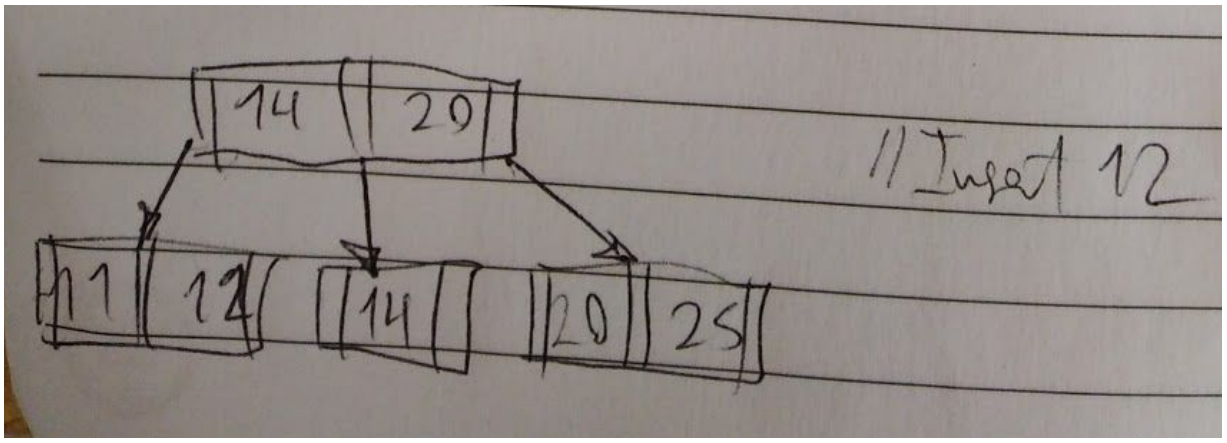
Insertar 20, 11, 14



Insertar 25

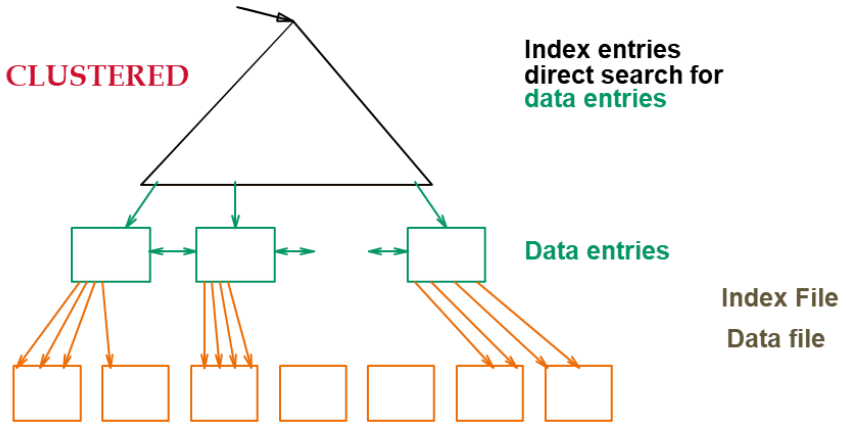


Insertar 12



P3. B+ Tree: Búsqueda

Proponga un algoritmo de búsqueda para B+ Tree cuando el índice es agrupado.




```
// Llamada base de la función
findRegister(register) {
    findRegister(register, topBucket);
}

// Llamada recursiva de la función
findRegister(register, bucket) {
    i = 0
    while (i < bucket.KeyList.size() AND register.key < bucket.KeyList[i]) {
```

```
            i++;
        }
        if (pointerToChildRegister[i].isNull()) {
            return bucket.RegisterList[i];
        }
        return findRegister(register, getBucket(bucket.KeyList[i]));
}
```